UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/799,351	03/12/2004	David Dehghan	14917.1363US01	6346

27488 7590 09/01/2009

MERCHANT & GOULD (MICROSOFT)
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
LE, MIRANDA	
ART UNIT	PAPER NUMBER
2159	
MAIL DATE	DELIVERY MODE
09/01/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/799,351	DEHGHAN ET AL.	
	Examiner	Art Unit	
	MIRANDA LE	2159	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *30 June 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7,9,10 and 30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7,9,10 and 30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 06/30/09 has been entered.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order

for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

Claims 1, 2, 3, 10, 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sakanishi (US Patent No. 6,678,888), in view of Melchione, Daniel Joseph (US Pub No. 20030200300), and further in view of Crudele et al. (US Patent No. 6,973,647).

As to claims 1, 30, Sakanishi teaches a software update distribution system/method for distributing a software update over a communication network for distribution to client computers, comprising:

a root update service node *(i.e. FIG. 15 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a recording medium, col. 10, lines 55-59)*; and

a plurality of child update service nodes operable to distribute software updates to client computers *(i.e. FIG. 17 is a diagram showing flows of information in determination of software to be distributed in a controlled system upon reception of a command making a request for software distribution from an upper-level system or upon reception of a command making a request for software distribution from the user of the controlled system. Receiving the command making a request for software distribution from the uppermost-level control system, the controlled system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level*

system. If the desired software file also does not exist in a management data base of the upper-level data base, the file is downloaded from the uppermost-level system. The controlled system retrieves management information to determine desired software. Then, the file of the desired software is downloaded from the upper-level system and the desired software is installed, col. 11, lines 20-38);

wherein the root update service node and the plurality of child update service nodes are organized in a hierarchical manner such that that the root update service node is a parent update service node to at least one child update service node, wherein each of the plurality of child update service nodes has a parent update service node, and wherein at least a first child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes *(i.e. FIG. 17 is a diagram showing flows of information in determination of software to be distributed in a controlled system upon reception of a command making a request for software distribution from an upper-level system or upon reception of a command making a request for software distribution from the user of the controlled system. Receiving the command making a request for software distribution from the uppermost-level control system, the controlled system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. If the desired software file also does not exist in a management data base of the upper-level data base, the file is downloaded from the uppermost-level system. The controlled system retrieves management information to*

determine desired software. Then, the file of the desired software is downloaded from the upper-level system and the desired software is installed, col. 11, lines 20-38); and

wherein the root update service node includes a first administration application programming interface (API) and first administration user interface, wherein the first administration API and first administration user interface are operable to receive from an administrator a set of rules for distributing software updates to at least some of the plurality of child update service nodes *(i.e. FIG. 13 is a diagram showing flows of information in determination of software to be distributed upon reception of a command making a request for software distribution in an uppermost-level system. As shown in the figure, the user enters a software-identifying ID (or a software name), a version of software to be distributed and information on a controlled system or information specifying a distribution target to the uppermost-level system. A plurality of pieces of software to be distributed and a plurality of destination targets may be entered. As the user enters a distribution command, management information is retrieved from management data bases to determine software to be distributed. A command making a request for software distribution is then created and issued to a lower-level system. Receiving the command making a request for software distribution, the lower-level system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. If the lower-level system is a control system, only a command making a request for software distribution is transmitted thereto from the upper-level system. In this example, the lower-level system is a controlled system.*

Thus, a distributed file is also transmitted thereto along with the distribution information. Receiving the command making a request for software distribution and a distributed file, the controlled system installs the software in accordance with the instruction, col. 10, lines 11-40); and

wherein at least the first child update service node includes a second administration API, separate from the first administration API, and a second administration user interface, separate from the first administration user interface, wherein the second administration API and second administration user interface are operable to receive a second set of rules for distributing software updates from the first child update service node to at least one other child update service node *(i.e. FIG. 14 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a file. Even if the software and the command making a request for software distribution are distributed by way of a management server, the file for storing the software and the command making a request for software distribution is transmitted to a lower-level system with the state of the file kept as it is. Receiving the distributed file including the command making a request for software distribution, the controlled system installs the software stored in the file in accordance with the instruction, col. 10, lines 41-54)*; and

wherein the root update service node obtains a software update from a software provider, and wherein each of the plurality of child update service nodes obtains the software update for distribution to the client computers by obtaining the software update

from its parent update service node *(i.e. FIG. 16 is a diagram showing flows of data in determination of software to be distributed upon reception of a command making a request for software distribution in an intermediate control system. The uppermost-level control system creates a software-identifying ID, a version and information on a controlled system in a simple combination and generates a command making a request for software distribution. Receiving the command making a request for software distribution from the uppermost-level control system, the intermediate control system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. Then, management information is retrieved from a management data base to determine software to be distributed. A command making a command making a request for software distribution is created and transmitted to a lower-level system along with the software to be distributed. If the lower-level system is a control system only a command making a request for software distribution is transmitted thereto from the upper-level system. In this example, the lower-level system is a controlled system. Thus, a distributed file is also transmitted thereto along with the distribution information. Receiving the command making a request for software distribution and distributed file, the controlled system installs the software In accordance with the command, col. 10, line 60 to col. 11, line 19).*

Sakanishi implicitly teaches "administration" as in Fig. 2.

Melchione expressly teaches an administration application programming interface (API) through which an administrator established rules for distributing software

updates from the update service node to its child update service nodes *(i.e. administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021]).*

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer and Melchione at the time the invention was made to modify the system of Mayer to include the limitations as taught by Melchione. One of ordinary skill in the art would be motivated to make this combination in order to represent some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider in view of Melchione, as doing so would give the added benefit of new releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator as taught by Melchione ([0021]).

Sakanishi implicitly teaches "a set of rule" as in Figs. 10, 11.

Mayer, Melchione do not state the term "a set of rules."

Crudele teaches a set of rules *(i.e. there is provided a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining respective modifications to said client processing system and at least one file required to implement said at least one modifying action, said tool comprising: a plurality of classes, each class corresponding to a respective type of action; means for reading said software package file and instantiating a class having attributes corresponding to the respective type of each of the at least one action of said software package file and setting the attributes of the at least one class according to the respective action definition in said software package file, means for executing a check method on at least one of each of said at least one class instances to determine if a deployment operation can be implemented in a specified first mode; means, responsive to a successful check, for executing a method on each of said at least one class instances in said first mode; and means, responsive to check failure of any class instance, for executing a method on each of said at least one class instances in a second less preferable mode, col. 2, lines 10-34).*

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione and Crudele at the time the invention was made to modify the system of Mayer, Melchione to include the limitations as taught by Crudele. One of ordinary skill in the art would be motivated to make this combination in order to provide a complete definition of the actions involved in a software distribution in view of Crudele (Abstract), as doing so would give the added benefit of that when a software package is

available to a target endpoint, an engine resident on the target can be instructed via the management agent to decode the software package from the file into memory and then to perform various software distribution operations including installing, removing and modifying the software and configuration of the endpoint as taught by Crudele (Abstract).

As per claim 2, Sakanishi, as combined, teaches the software update distribution system of Claim 1, wherein the root update service node comprises:

an update store for storing software updates *(i.e. FIG. 15 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a recording medium, col. 10, lines 55-59)*;

an update web service through which the root update service node distribute software updates to its child update service nodes over the communication network *(i.e. FIG. 15 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a recording medium, col. 10, lines 55-59)*; and

a software provider interface through which a software provider submits its software update over the communication network to the root update distribution node *(i.e. FIG. 15 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software*

distribution, which are stored in a recording medium, col. 10, lines 55-59).

As per claim 3, Sakanishi, as combined, teaches the software update distribution system of Claim 2, wherein each of the plurality of child update service nodes comprises:

an update store for storing software updates *(i.e. FIG. 14 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a file. Even if the software and the command making a request for software distribution are distributed by way of a management server, the file for storing the software and the command making a request for software distribution is transmitted to a lower-level system with the state of the file kept as it is. Receiving the distributed file including the command making a request for software distribution, the controlled system installs the software stored in the file in accordance with the instruction, col. 10, lines 41-54)*;

an update web service through which the child update service node obtains software updates form its parent update service node over the communication network, and through which the child update service node distributes software updates to its update service nodes over the communication network *(i.e. FIG. 14 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a file. Even if the software and the command making a request for software*

distribution are distributed by way of a management server, the file for storing the software and the command making a request for software distribution is transmitted to a lower-level system with the state of the file kept as it is. Receiving the distributed file including the command making a request for software distribution, the controlled system installs the software stored in the file in accordance with the instruction, col. 10, lines 41-54);

a child update module for determining which software updates are available to be distributed to its child update service nodes *(i.e. FIG. 14 is a diagram showing an image of distributing software determined by an uppermost-level system for distribution and a command making a request for software distribution, which are stored in a file. Even if the software and the command making a request for software distribution are distributed by way of a management server, the file for storing the software and the command making a request for software distribution is transmitted to a lower-level system with the state of the file kept as it is. Receiving the distributed file including the command making a request for software distribution, the controlled system installs the software stored in the file in accordance with the instruction, col. 10, lines 41-54).*

Crudele teaches a set of rules *(i.e. According to the present invention there is provided a software deployment tool cooperable with a software package including a software package file incorporating at least one action defining respective modifications to said client processing system and at least one file required to implement said at least one modifying action, col. 2, lines 10-34)*.

As per claim 10, Melchione, as combined, teaches the software update distribution system of Claim 3, wherein the first child update service node may be selectively configured to periodically obtain available software updates from the root update service node *(i.e. The nodes can include agent software that periodically communicates with the data center 712, [0078])*.

Claims 4-7, 9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sakanishi (US Patent No. 6,678,888), in view of Melchione, Daniel Joseph (US Pub No. 20030200300), and Crudele et al. (US Patent No. 6,973,647), and further in view of Donohue (US Patent No. 6,199,204).

As per claim 4, Sakanishi teaches the software update distribution system of Claim 3, wherein each of the plurality of child update service nodes further comprises:

a reporting module for generating and sending update activity reports to the parent update service node *(i.e. In this embodiment, control systems can also be arranged in a distributed configuration or a multi-stage configuration. In a distributed configuration or a multi-stage configuration comprising a plurality of control systems, a control system cataloging a new piece of software communicates the new piece of software and information on software management to other control systems at the end of the cataloging process so that all the control systems manage the same pieces of software. As an alternative conceivable method, instead of communicating the newly cataloged piece of software, information on the control system cataloging the new piece of software is added to the transmitted information on management, and the new piece*

of software itself is managed by the control system cataloging the new piece of software, col. 5, lines 33-47); and

a client update module for distributing software updates to client computers *(i.e. In this embodiment, control systems can also be arranged in a distributed configuration or a multi-stage configuration. In a distributed configuration or a multi-stage configuration comprising a plurality of control systems, a control system cataloging a new piece of software communicates the new piece of software and information on software management to other control systems at the end of the cataloging process so that all the control systems manage the same pieces of software. As an alternative conceivable method, instead of communicating the newly cataloged piece of software, information on the control system cataloging the new piece of software is added to the transmitted information on management, and the new piece of software itself is managed by the control system cataloging the new piece of software, col. 5, lines 33-47)*

Mayer, Melchione, Crudele do not seem to explicitly teach:

an authentication and authorization module for determining whether an update service node is authorized to obtain software updates from the child update service node.

Donohue teaches:

an authentication and authorization module for determining whether an update service node is authorized to obtain software updates from the child update service node *(i.e. The updater component preferably also includes a mechanism for verifying*

the authenticity of downloaded software, using cryptographic algorithms. This avoids the need for dedicated, password-protected or otherwise protected software resource repository sites. The software resources can be anywhere on the network as long as they are correctly named and or posted to the network search engines, col. 5, lines 36-42).

It would have been obvious to one of ordinary skill of the art having the teaching of Mayer, Melchione, Crudele, Donohue at the time the invention was made to modify the system of Mayer, Melchione, Crudele to include the limitations as taught by Donohue. One of ordinary skill in the art would be motivated to make this combination in order to verify the authenticity of downloaded software in view of Donohue (col. 5, lines 36-42), as doing so would give the added benefit of protecting software resource repository sites as taught by Donohue (col. 5, lines 36-42).

As per claim 5, Sakanishi, as combined, teaches the software update distribution system of Claim 4, wherein the update store comprises an update content store in which the update payload for the software update is stored, and an update information store in which update metadata for the software update is stored *(i.e. FIG. 17 is a diagram showing flows of information in determination of software to be distributed in a controlled system upon reception of a command making a request for software distribution from an upper-level system or upon reception of a command making a request for software distribution from the user of the controlled system. Receiving the command making a request for software distribution from the uppermost-*

level control system, the controlled system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. If the desired software file also does not exist in a management data base of the upper-level data base, the file is downloaded from the uppermost-level system. The controlled system retrieves management information to determine desired software. Then, the file of the desired software is downloaded from the upper-level system and the desired software is installed, col. 11, lines 20-38).

As per claim 6, Sakanishi, as combined, teaches the software update distribution system of Claim 5, wherein the first child update service node obtains the software update from the root update service node by obtaining update metadata for the software update form the root update service node, and separately obtaining the update payload for the software update from the root update service node *(i.e. FIG. 17 is a diagram showing flows of information in determination of software to be distributed in a controlled system upon reception of a command making a request for software distribution from an upper-level system or upon reception of a command making a request for software distribution from the user of the controlled system. Receiving the command making a request for software distribution from the uppermost-level control system, the controlled system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. If the desired software*

file also does not exist in a management data base of the upper-level data base, the file is downloaded from the uppermost-level system. The controlled system retrieves management information to determine desired software. Then, the file of the desired software is downloaded from the upper-level system and the desired software is installed, col. 11, lines 20-38).

As per claim 7, Sakanishi, as combined, teaches the software update distribution system of Claim 6, wherein the first child update service node obtains the update payload for the software update from the root update service node in a just-in-time fashion *(i.e. FIG. 17 is a diagram showing flows of information in determination of software to be distributed in a controlled system upon reception of a command making a request for software distribution from an upper-level system or upon reception of a command making a request for software distribution from the user of the controlled system. Receiving the command making a request for software distribution from the uppermost-level control system, the controlled system checks whether or not the desired software file exists in a management data base. If the desired software file does not exist in a management data base, the file is downloaded from the upper-level system. If the desired software file also does not exist in a management data base of the upper-level data base, the file is downloaded from the uppermost-level system. The controlled system retrieves management information to determine desired software. Then, the file of the desired software is downloaded from the upper-level system and the desired software is installed, col. 11, lines 20-38).*

As per claim 9, Melchione, as combined, teaches the software update distribution system of Claim 8, wherein the root update service node further comprises a client update module for distributing software updates to client computers *(i.e. In one case, administered software may be downloaded and run at a customer computer. In other cases, administered software represents some portion of administered software in processing communication with other administered software residing on other customer computers or at the provider. In another case, administered software may run at the provider with only a thin client (e.g., a web browser) running at a customer computer (e.g., displaying administered software output). All these cases, represent examples of provider-hosted application services, [0020]; In certain described examples, the software being administered is anti-virus software. New releases of anti-virus software can be automatically provided according to the software administration directives set by a customer administrator. In some case, administered software includes an agent that polls the data center to obtain administered software updates, [0021])*.

Response to Arguments

Applicant's arguments with respect to claims 1-7, 9-10, 30 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Miranda Le whose telephone number is (571) 272-4112. The examiner can normally be reached on Monday through Friday from 10:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James K. Trujillo, can be reached at (571) 272-3677. The fax number to this Art Unit is (571)-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (571) 272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Miranda Le/

Primary Examiner, Art Unit 2159